Utilisation du logiciel S-PLUS©

J. Thioulouse

1 - Utilisation de base sur station Unix

Résumé

Cette fiche présente les notions de base nécessaires à l'utilisation du logiciel S-PLUS 5 sur station de travail Unix : connexion au serveur, configuration de la session, lancement du logiciel, édition des commandes, lecture / écriture de fichiers de données et création de fonctions en langage S.

1 - Introduction

S-PLUS est un logiciel de calcul statistique basé sur un environnement orienté objet et sur le langage de programmation S. Il est disponible sur PC avec Microsoft Windows 95 / NT et sur stations Unix avec X-Windows (SUN Sparc, IBM RS/6000, Hewlett-Packard 9000-700, Silicon Graphics IRIS, DECStation, DEC Alpha/OSF). Il n'existe pas de version Macintosh.

La version utilisée pour Unix est Splus 5.0, celle pour PC est Splus 4.5. Les fonctionnalités statistiques sont globalement identiques, mais l'interface utilisateur est en mode "ligne de commande" dans la version Unix, alors qu'elle est en mode graphique dans la version PC. Les fonctionnalités liées à l'interface utilisateur sont bien plus développées dans la version PC (système d'aide, lancement des commandes, manipulations des objets et des fichiers, édition des graphiques, etc.) Les principaux avantages de la version Unix sont surtout la possibilité de traiter de très grands jeux de données (plusieurs Go), l'import-export de données provenant des SGBD professionnels (Oracle, Sybase, Informix), la facilité de développement de fonctions en langage C et en FORTRAN, et l'utilisation centralisée avec licence multi-utilisateur.

La version Unix est disponible sur les stations Sun du laboratoire de Biométrie (pbil, acnuc et biomserv, avec une licence pour 50 utilisateurs). Elle peut être utilisée à partir de n'importe quel ordinateur connecté au réseau Internet (Sun, Mac, PC, terminal X).

L'utilisation à partir d'un micro-ordinateur Mac ou PC nécessite un logiciel de connexion au serveur Sun en mode X-Windows (émulation d'un terminal X), ou le logiciel VNC (http://www.uk.research.att.com/vnc/), qui fontionne avec un serveur X virtuel.

L'utilisation de la version de Splus pour Unix implique une connaissanceminimale du système Unix, nécessaire pour se connecter au serveur, configurer la session de travail et manipuler les fichiers de données.

2 - Connexion

La connexion à une station de travail Sun nécessite un nom d'utilisateur et un mot de passe. Pour se connecter sur les stations Sun de la salle de TP, les noms d'utilisateurs de chaque étudiant sont donnés en annexe à la fin de ce document et les mots de passe sont constitués par le nom d'utilisateur suivi de ••••.

Une fois connecté sur la station de travail, il faut se connecter au serveur du laboratoire de Biométrie sur lequel se trouve le logiciel Splus (pbil.univ-lyon1.fr). Les étudiants de DEA peuvent se servir du nom d'utilisateur "dea" et du mot de passe ••••••. Les autres utilisateurs peuvent se servir de leur propre compte.

Avant d'établir la connexion avec le serveur, il faut autoriser l'affichage des fenêtres du serveur sur son propre poste. Ceci est réalisé au moyen de la commande **xhost** (les caractères en gras sont ceux tapés par l'utilisateur).

```
user% xhost pbil.univ-lyon1.fr
```

Cette manoeuvre est indispensable et rien ne marchera si elle n'est pas effectuée avant la connexion aux serveurs du laboratoire de Biométrie. Le nom du serveur "pbil.univ-lyon1.fr" peut être remplacé par le nom abrégé "pbil" quand on se trouve sur le campus de la Doua à Lyon.

La connexion au serveur pbil se fait ensuite en lançant la commande telnet :

```
user%  telnet pbil.univ-lyon1.fr
Trying 134.214.94.219...
Connected to pbil.univ-lyon1.fr.
Escape character is '^]'.


SunOS 5.6
```

```
login: dea
Password:••••••
Last login: Mon Nov 15 10:46:50 from cripsun3.univ-ly
Sun Microsystems Inc.   SunOS 5.6       Generic August 1997
pbil:dea
```

Le prompt "`pbil:dea`" permet de vérifier la réussite de l'opération. Une fois la connexion réalisée, la première chose à faire consiste à indiquer sur quel poste on travaille et sur quel écran les fenêtres doivent s'afficher. La commande à utiliser pour ça est la suivante :

```
pbil:dea setenv DISPLAY cripsun9:0
```

`cripsun9` est à remplacer par le nom de la station sur laquelle on travaille (**cripsunx**, x étant le numéro de votre poste dans la salle). Verifier l'état de la connection avec la commande `xlogo`, qui doit afficher le logo X-Window dans une fenêtre. Ne pas oublier de terminer l'execution de cette commande (taper control-C, ou utiliser l'option Quit du menu local de la fenêtre xlogo).

Dans le cas d'une connexion à partir d'un terminal X, les commandes xhost et setenv DISPLAY sont en principe automatiques. Dans le cas d'une connexion à partir d'un micro-ordinateur Mac ou PC, il convient de vérifier que le logiciel d'émulation X est bien configuré pour autoriser l'affichage des fenêtres X-Window provenant du serveur pbil.univ-lyon1.fr. En fonction du logiciel utilisé, il peut également être nécessaire d'établir une session telnet avec un autre logiciel et d'utiliser la commande setenv DISPLAY. Il est alors nécessaire de connaître le nom de la machine sur laquelle on travaille.

3 - Configuration

Splus fait intervenir la notion de dossier de travail. Il s'agit d'un répertoire classique dont la seule particularité est la nécessité de la présence d'un sous-répertoire intitulé ".Data" (attention aux minuscules/majuscules !) Sous Unix, les fichiers et les répertoires dont le nom commence par un point (".") sont dits "invisibles". Cela signifie simplement qu'ils ne sont pas affichés lors de l'utilisation de la commande "ls" qui permet de voir la liste des fichiers. Pour les afficher, il faut utiliser la commande "ls" avec l'option "-a" (taper au clavier "ls -a").

Le dossier de travail Splus contiendra par exemple les fichiers de données, les fonctions écrites en langage S par l'utilisateur, ou les fichiers postscript issus d'une demande impression. Le répertoire .Data contiendra les objets crées par le logiciel Splus lors de l'utilisation. Ces objets sont stockés dans des fichiers binaires qui ne sont pas utilisables en dehors de Splus.

La création d'un dossier de travail se fait donc de la façon suivante :

```
pbil:dea mkdir Splus        // création du répertoire de travail
pbil:dea cd Splus                 // changement de répertoire
pbil:dea mkdir .Data        // création du répertoire .Data
pbil:dea ls
pbil:dea ls -a
./          ../         .Data/
pbil:dea pwd                      // affichage du répertoire courant
/mnt/users/dea/xxxxxx/Splus
```

La commande **mkdir** permet de créer un répertoire.

La commande **cd** permet de changer de répertoire. Le répertoire special "**.**" représente le répertoire courant. Le répertoire spécial "**..**" représente le répertoire de niveau immédiatement supérieur.

La commande **ls** permet d'afficher la liste des fichiers du répertoire courant.

La commande **pwd** permet d'afficher le répertoire courant.

Chaque étudiant de DEA possède sur la machine pbil un repertoire, situé dans le répertoire /mnt/users/dea.

3.1 - Création d'un script de lancement

Il faut ensuite créer un script de lancement du logiciel Splus. Ce script, que nous appellerons "StartSP", permettra d'indiquer à Splus le nom du dossier de travail, ainsi que le nom du fichier contenant l'historique des commandes exécutées dans Splus. Cet historique est très pratique car il permet de rappeler automatiquement et d'éditer les commandes qu'on a déjà tapé.

Création du script de lancement de Splus :

```
pbil:dea pwd
/mnt/users/dea/xxxxxx/Splus
pbil:dea cat > StartSP
#!/bin/csh
setenv S_WORK /mnt/users/dea/xxxxxx/Splus
setenv S_CLHISTFILE /mnt/users/dea/xxxxxx/Splus/.Splus_history
Splus5 -e
pbil:dea
```

Terminer la création du script en tapant control-D (Touche Controle + lettre D). Ne pas oublier de remplacer le chiffre 9 dans les commandes ci-dessus (ccmitp9) par le numéro de votre station.

La commande **cat** permet de créer un fichier (intitulé ici "StartSP"). L'opérateur de redirection ">" permet d'envoyer dans ce fichier les lignes de texte tapées par l'utilisateur à la suite de la commande cat. Il est également possible d'utiliser l'editeur de texte "**vi**", qui possède une interface utilisateur en mode commande, ou l'editeur de texte Sun "**textedit**", qui possède une interface utilisateur graphique.

La ligne #!/bin/csh permet d'indiquer quel doit être l'interprèteur de commandes utilisé pour exécuter le script (ici le c-shell).

Les deux commandes suivantes permettent de spécifier le nom du dossier de travail (/mnt/users/dea/xxxxxx/Splus) dans la variable d'environnement S_WORK, et le nom du fichier historique (/mnt/users/dea/xxxxxx/Splus/.Splus_history) dans la variable d'environnement S_CLHISTFILE.

La commande "Splus5 -e" lance l'exécution du logiciel Splus avec les paramètres requis. La création du script se termine en tapant le caractère contrôle-D. Il faut autoriser l'exécution de ce script avec la commande chmod :

```
pbil:dea chmod u+x Splus
```

Vérifier qu'on se trouve bien dans le bon répertoire (commande pwd), que le dossier de travail est complet (commande "ls -a"), et que le script Splus est bien executable (commande "ls -la", qui affiche les propriétés rwx des fichiers) :

```
pbil:dea pwd
/mnt/users/dea/xxxxxx/Splus
pbil:dea ls
StartSP*
pbil:dea ls -a
./          ../         .Data/      StartSP*
pbil:dea ls -la
total 224
drwxr-xr-x   3 dea      staff        512 Oct 27 17:05 ./
drwxr-xr-x  70 dea      staff       3072 Oct 27 17:14 ../
drwxr-xr-x   2 dea      staff        512 Oct 28  1998 .Data/
-rwxrwxrwx   1 dea      staff        120 Oct 27 16:21 Splus*
pbil:dea
```

Le fichier .Splus_history n'existe pas encore et sera créé lors de la première utilisation de Splus. L'étoile placée après le nom du fichier Splus indique qu'il s'agit d'un

script exécutable. le caractère "/" placé après le répertoire ".Data" indique qu'il s'agit d'un répertoire et non pas d'un fichier. "." et ".." désignent respectivement le répertoire courant et le répertoire de niveau supérieur. La commande "**cd ..**" remonte d'un niveau, et "**cd .**" ne fait rien du tout (cd = change directory).

Lancer l'exécution du script :

```
pbil:dea StartSP
S-PLUS : Copyright (c) 1988, 1998 MathSoft, Inc.
S : Copyright Lucent Technologies, Inc.
Version 5.0 Release 3 for Sun SPARC, SunOS 5.5 : 1998
Working data will be in /mnt/users/dea/xxxxxx/Splus
>
```

Le prompt ">" indique qu'on peut commencer à taper des commandes. Vérifier que le dossier de travail (Working data) est correct.

Quitter Splus avec la fonction **q()**.

```
> q()
pbil:dea
```

Il ne faut surtout pas quitter Splus d'une autre façon car il reste alors des process en mémoire, qu'il convient de terminer proprement ensuite (cf ci-dessous).

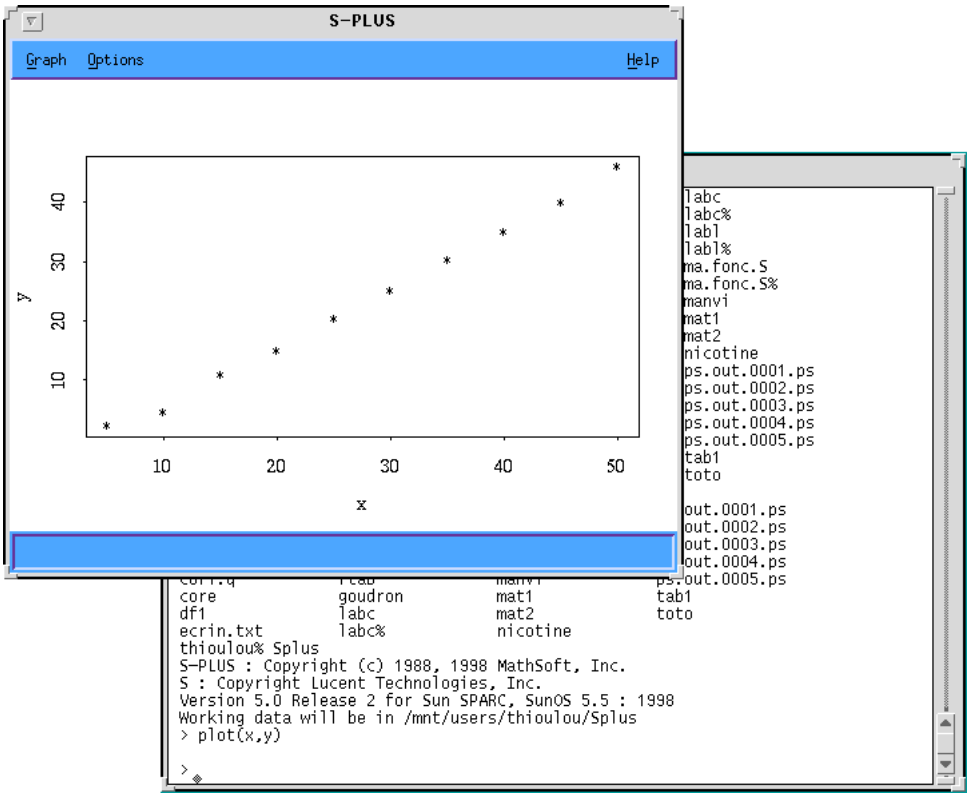
3.2 - Ouverture d'une fenêtre graphique

Ouverture et fermeture d'une fenêtre graphique : la fonction **motif()** et la fonction **dev.off()** :

```
pbil:dea StartSP
S-PLUS : Copyright (c) 1988, 1998 MathSoft, Inc.
S : Copyright Lucent Technologies, Inc.
Version 5.0 Release 3 for Sun SPARC, SunOS 5.5 : 1998
Working data will be in /mnt/users/dea/xxxxxx/Splus
> motif()
```

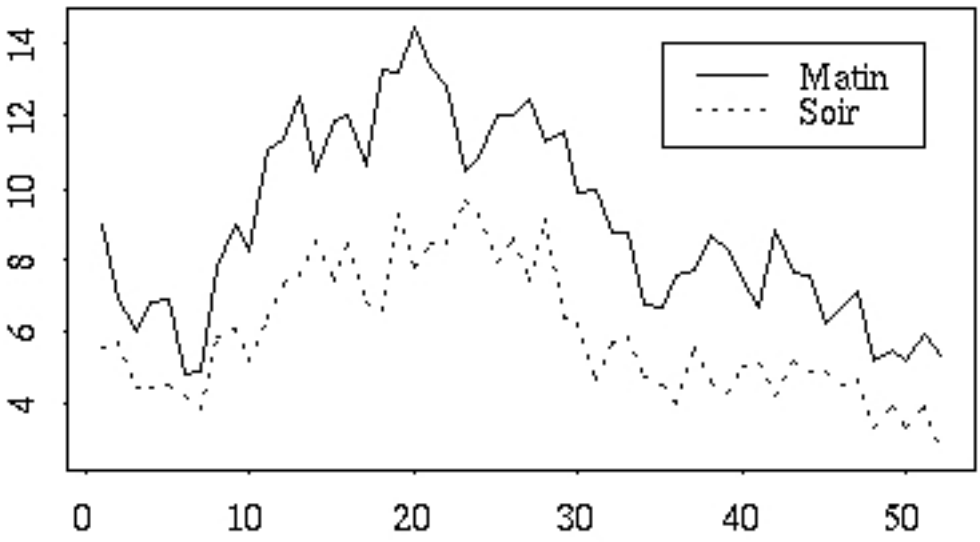
Une fenêtre graphique redimensionable s'affiche à l'écran. Pour la refermer, il faut obligatoirement utiliser la fonction dev.off() et **non pas** le menu associé à la fenêtre :

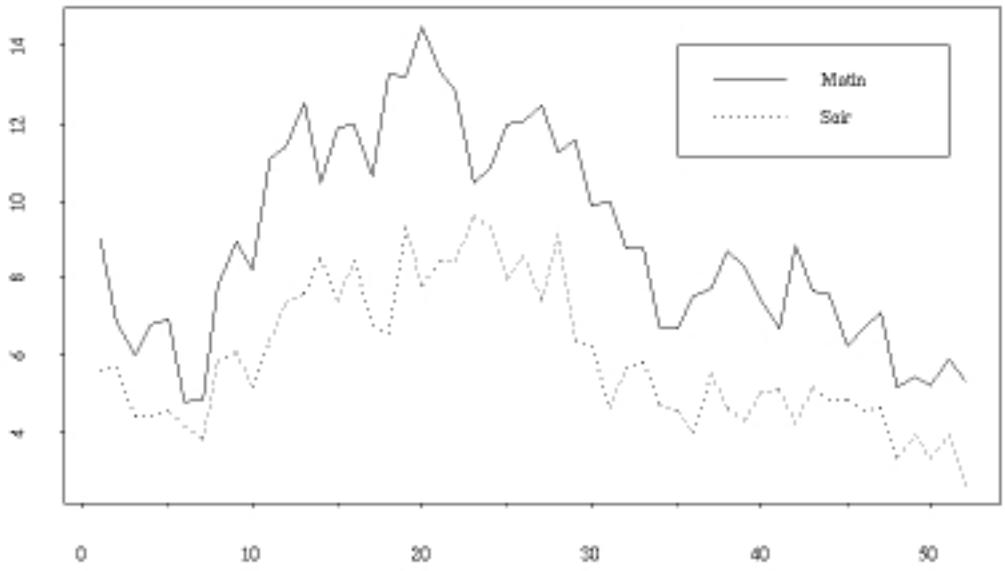
```
> dev.off()
```



La fonction motif() accepte des arguments, en particulier la dimension et la position de la fenêtre : **`motif("-geometry 500x400+20+20")`**

La fenêtre graphique possède un menu "Help" qui explique son fonctionnement, un menu "Graph" qui permet de retracer le graphique, de l'imprimer, ou de l'enregistrer dans un fichier Postscript, et un menu "Options" qui permet de paramètrer l'impression ainsi que les couleur d'affichage à l'écran (en particulier noir / blanc ou blanc / noir).





Les graphiques obtenus peuvent être récupérés en format Postscript, ou en format bitmap par copie d'écran. Dans le second cas, pour obtenir des graphiques de bonne qualité, il convient d'utiliser une fenêtre de grande taille et de diminuer ensuite la taille de l'image, tout en conservant sa résolution. Ceci peut être réalisé dans Microsoft Word de la façon suivante : passer en mode d'affichage normal (menu "Affiche", option "Normal"), cliquer une fois sur l'image pour la sélectionner, appuyer sur la touche majuscule et la maintenir enfoncée, cliquer avec la souris dans le petit carré noir situé dans le coin inférieur droit de l'image en maintenant le bouton appuyé, et déplacer la souris jusqu'à obtenir la taille de l'image souhaitée. Le facteur de rédction s'affiche dans le coin inférieur gauche de la fenêtre de Word : une réduction de 50% donne en général de bons résultats (passage de 72 à 144 ppp).

Les deux figures ci-dessus ont été tracées dans des fenêtres de dimension différentes : 500 x 400 pour la première et 900 x 650 pour la seconde. La première a été copiée et collée sans transformation dans le document Word, alors que la seconde a reçu une réduction de 50%. Il peut être nécessaire d'augmenter la taille de la police de caractères des légendes des axes avec le paramètre cex de la fonction par() pour conserver des valeurs lisibles.

3.3 - Affichage et modification des options

La fonction **options()** sans arguments affiche la liste des options et leur valeur courante :

```
> options()
```

Pour modifier une option, il suffit d'appeler la fonction option() avec l'option désirée et la valeur à lui affecter :

```
> pi
[1] 3.141593
> options(digits=3)
> pi
[1] 3.14
>
```

On peut demander la valeur d'une option particulière :

```
> options("digits")
$digits:
[1] 3
```

3.4 - Système d'aide en ligne

Taper un point d'interrogation pour obtenir de l'aide sur le système d'aide en ligne. Il est possible d'obtenir de l'aide sur toutes les fonctions de Splus :

```
> ?lm
Fit Linear Regression Model
DESCRIPTION:
       Returns an object of class "lm" or "mlm" that represents a
       fit of a linear model.
USAGE:
       lm(formula, data=<<see below>>, weights=<<see below>>,
       subset=<<see below>>, na.action=na.fail, method="qr",
       model=F, x=F, y=F, contrasts=NULL, ...)

REQUIRED ARGUMENTS:
formula:    a formula object, with the response on the left of  a
       ~  operator,  and  the terms, separated by + operators, on
       the right.

OPTIONAL ARGUMENTS:
data:     a data.frame in which to interpret the variables  named
       in the formula, or in the subset and the weights argument.
       If this is missing, then  the  variables  in  the  formula
       should  be  on the search list.  This may also be a single
       number to handle  some  special  cases  -  see  below  for
       details.
weights:    vector  of  observation  weights;  if  supplied,  the
       algorithm   fits   to   minimize   the   sum   of   the   weights
--More--(12%)
```

L'affichage se fait page par page et il faut taper sur la barre d'**espace** pour passer d'une page à la suivante (pagination par la commande Unix "**more**"). Si on veut sortir de l'aide avant la fin du défilement, il suffit de taper sur la touche "**q**" (quit).

3.5 - Création d'une fonction à exécution automatique

Il est possible de créer une fonction qui s'exécute automatiquement au lancement de Splus. Cette fonction doit simplement s'appeler ".First". On peut par exemple utiliser une fonction qui crée une fenêtre graphique de dimensions choisies et fixe le nombre de chiffres significatifs à 3 :

```
> .First<-function(){
+ motif("-geometry 500x400+20+20")
+ options(digits = 3)
+ }
```

Pour afficher le contenu de cette fonction, il suffit de taper son nom (sans parenthèses) :

```
> .First
function()
{
      motif("-geometry 500x400+20+20")
      options(digits = 3)
}
>
```

Pour lancer l'exécution de la fonction, il faut taper son nom, suivi de la liste des arguments entre parenthèses. Une paire de parenthèses, même vides, est toujours nécessaire. En effet, si on ne met pas de parenthèses, Splus affiche le contenu de la focntion au lieu de lancer son exécution (Cf paragraphe précédent).

3.6 - Ménage

En cas de plantage dans Splus, ou en cas de sortie de Splus d'une autre façon que par la fonction **q()**, il convient de terminer les process qui restent éventuellement en cours d'exécution avant de lancer une nouvelle session Splus. Utiliser la commande "ps -ef | grep plus" pour afficher tous les process Splus. Il y a en général au moins 6 process en cours d'exécution pour chaque session :

```
pbil:dea ps -ef | grep plus
dea  4504  4499  0 18:39:24 pts/10   0:00 /opt/splus5.0/cmd/cled_jctl /opt/splus5.0/cmd/NEW
dea  4505  4504  1 18:39:24 pts/10   0:01 /opt/splus5.0/cmd/Sqpe
dea  4536  8017  0 18:39:32 pts/2    0:00 grep plus
dea  4498  4339  0 18:39:24 pts/3    0:00 /bin/csh Splus
dea  4521  4505  0 18:39:26 pts/10   0:00 /opt/splus5.0/splus/x/dev.motif
dea  4512  4505  0 18:39:25 pts/10   0:00 /opt/splus5.0/adm/lic/cmd/Slmclient
dea  4499  4498  0 18:39:24 pts/3    0:00 /opt/splus5.0/cmd/cled /opt/splus5.0/cmd/cled_jctl /opt/splus5.0/cmd/NEW
pbil:dea
pbil:dea kill 4504
```

Repérer la ligne sur laquelle se trouve la chaîne "cled_jctl", et utiliser la commande Unix **kill** avec le numéro du process correspondant à cette ligne (il s'agit du premier nombre de la ligne).

```
pbil:dea kill 4499
```

Attention : ne pas effectuer cette opération pendant les TP car vous risquez de terminer les sessions de vos camarades.

4 - Édition

L'interface utilisateur de Splus 5 est en mode ligne de commande, et il est donc utile de pouvoir éditer facilement les lignes de commandes déjà tapées, afin de corriger des erreurs ou d'éviter d'avoir à taper plusieurs fois des commandes complexes. L'édition des commandes Splus peut faire appel aux éditeurs **vi** ou **emacs**, l'éditeur par défaut étant **vi**. Notez qu'il est nécessaire de lancer l'exécution de Splus avec l'option "-e" ("Splus5 -e") pour bénéficier des possibilités d'édition des commandes. Pour utiliser l'éditeur emacs au lieu de vi, il faut de plus que la variable d'environnement EDITOR contienne le chaîne de caractères "emacs" : utiliser l'ordre "`setenv EDITOR emacs`" avant le lancement de Splus.

4.1 - Touches d'édition

Voici la liste des touches utilisables pour l'édition des commandes (Ctrl = touche contrôle, Esc = touche escape, Maj = touche majuscule) :

Action	**Touche vi**
1 ligne en arrière	k
1 ligne en avant	j
1 car. en arrière	h
1 car. en avant	l

supprimer 1 car.	x
supprimer 1 mot	dw
supprimer 1 ligne	dd
rechercher	/
1 mot en arrière	b
1 mot en avant	w
début de ligne	Maj 6
fin de ligne	Maj 4

Il faut toujours d'abord entrer dans le mode d'édition en tapant sur la touche **escape** (esc). Pour insérer des caractères il faut déplacer le curseur à la position voulue avec les touches de déplacement (h, j, k, l), puis taper sur la touche **i** (insertion). On peut alors taper les caractères à insérer, puis taper **escape** à nouveau pour sortir du mode d'insertion.

Exemple : pour tracer un graphe (x,y), vous avez tapé la commande plot(x,y) avec une faute de frappe qui provoque un message d'erreur :

```
> plto(x,y)
Problem: Couldn't find a function definition for "plto"
```

Pour corriger cette erreur, il faut taper la série de touches suivante :

Esc	passage en mode édition
k	rappel de la commande précédente, curseur en début de ligne
l	avancer de 1 caractère
l	avancer de 1 caractère
x	supprimer le caractère "t"
l	avancer de 1 caractère (pour sauter le "o")
i	passer en mode insertion
t	taper le caractère "t"

L'utilisation de la touche **escape** pour sortir dun mode d'insertion n'est pas obligatoire si il n'y a pas d'autres corrections à effectuer. Il suffit dans ce cas de taper un retour-charriot pour valider la commande corrigée.

Remarque : lorsqu'on a procédé à une manipulation de fenêtres (passage dans la fenêtre graphique, déplacement ou utilisation d'une autre fenêtre, etc.), il est nécessaire de taper un

retour-charriot dans la fenêtre Splus avant de taper le caractère **escape** pour passer en mode édition.

4.2 - Historique des commandes

La fonction **history()** permet de rappeler les commandes passées. Utilisée sans argument elle présente la liste des 10 commandes précédentes :

```
> history()
1: q()
2: q()
3: plot(x, y)
4: ?xrmresource
5: q()
6: q()
7: plot(x, y)
8: q()
9: ?history
10: ?history
Selection: 3
> plot(x, y)
```

L'historique ne se limite pas à la session courante. Il s'étend à toutes les sessions successives effectuées avec le même fichier d'historique des commandes (par défaut le fichier ".Splus_history") spécifié avec la variable d'environnement S_CLHISTFILE. La fonction history peut également être utilisée pour effectuer des recherches dans les commandes passées :

```
> history("plot")
1: plot(lmx)
2: plot(anova(lmx))
3: plot(X1, X2, data = df1)
4: plot(df1$X1, df1$X2)
5: ?plot
6: ?plot
7: plot(x, y)
8: plot(x, y)
9: plot(x, y)
10: plot(x, y)
Selection:
```

Les résultats de la fonction history peuvent aussi être enregistrés dans un fichier et utilisés pour créer des fonctions utilisateurs.

5 - Lectures / Ecritures

Les fonctions d'entrée-sortie sont de deux types : celles opérant sur des données non structurées et celle opérant sur des tableaux (data.frames). Les premières (scan et cat) permettent de lire et d'écrire des fichiers texte de façon très simple. Les secondes permettent de manipuler des tableaux en conservant la structure en lignes/colonnes et les labels associés.

5.1 - scan et cat

La fonction scan peut être utilisée pour saisir des données au clavier mais aussi pour les lire dans un fichier :

```
> mat<-matrix(scan("mat1"),nrow=13,byrow=T)
```

Il est possible de ranger les valeurs dans une matrice avec la fonction matrix(). Dans ce cas, le nombre de lignes doit être spécifié (nrow=13), et il faut préciser que la lecture doit se faire ligne par ligne (byrow=T) car l'ordre "naturel" dans Splus est colonne par colonne.

La fonction cat permet d'effectuer l'opération inverse :

```
> cat(mat,file="mat2",fill=T)
```

L'option fill=T indique qu'il faut rajouter des retours-charriots a la fin des lignes. Sinon toutes les valeurs sont simplement écrites les unes à la suite des autres.

5.2 - read.table et write.table

Les fonctions read.table et write.table sont plus évoluées et permettent de manipuler des tableaux avec des labels de lignes et de colonnes. L'écriture se fait avec write.table en indiquant le tableau à écrire et le nom du fichier de sortie :

```
> write.table(Class1, file="Class1")
```

La lecture se fait avec read.table en indiquant plusieurs options :

```
> Class2<-read.table(file="Class1",header=T,",",1)
```

• header=T signifie qu'il existe un en-tête dans le fichier, contenant les noms des colonnes. Cet en-tête est écrit automatiquement par write.table.

• "," est le séparateur utilisé par la fonction write.table. On peut utiliser d'autres séparateurs pour les fichiers provenant d'ailleurs, et par exemple "\t" pour indiquer le caractère de tabulation

• 1 est le numéro de la colonne contenant les noms des lignes (écrit automatiquement par write.table). Sinon utiliser row.names = NULL.

Exemple : création et lecture/écriture d'un tableau 3 lignes x 2 colonnes.

```
> lablig<-scan()
1: l1
2: l2
3: l3
4:
> labcol<-scan()
1: c1
2: c2
3:
> tab1<-matrix(scan(),nrow=3,ncol=2,byrow=T,dimnames=list(
lablig,labcol))
1: 1
2: 2
3: 3
4: 4
5: 5
6: 6
7:
> tab1
   c1 c2
l1  1  2
l2  3  4
l3  5  6
> write.table(tab1,file="tab1")
> tab2<-read.table(file="tab1", header=T, ",", 1)
> tab2
   c1 c2
l1  1  2
l2  3  4
l3  5  6
> q()
```

Le fichier tab1 contient le tableau et les labels des lignes et des colonnes avec "row.names" comme label de la colonnes des labels des lignes :

```
pbil:dea more tab1
row.names,c1,c2
l1,1,2
l2,3,4
l3,5,6
pbil:dea
```

6 - Écriture de fonctions

La création d'une fonction simple peut se faire simplement en tapant sa définition au clavier dans Splus :

```
> .First<-function(){
+ motif("-geometry 500x400+20+20")
+ options(digits = 3)
+ }
```

Le fait de taper une accolade ouvrante entame le mode de continuation, symbolisé par le prompt "+". Ce mode est terminé par la frappe d'une accolade fermante, qui termine également la définition de la fonction.

Pour les fonctions un peu complexes, il est nécessaire de passer par l'intermédiaire d'un fichier texte, qu'on peut éditer avec un éditeur comme vi (mode commande) ou textedit (mode graphique). Il est alors nécessaire de pouvoir lire et écrire les fonctions dans un fichier externe. Ceci est réalisé grâce aux fonctions dput et source.

La fonction **dput** permet d'écrire un fonction S dans un fichier texte :

```
> dput(fonc1, file="fonc1.S")
```

La fonction **source** permet de relire une fonction S à partir d'un fichier texte :

```
> fonc1<-source("fonc1.S")
```

Exemple : créer une fonction qui renvoie le produit de ses deux arguments. La fonction est d'abord tapée dans un fichier texte sous Unix (commande **cat**, ou éditeur de texte au choix) :

```
pbil:dea cat > fonc1.S
function(x1, y1)
{
      z1 = x1 * y1
      return(z1)
}
^D
pbil:dea
```

Relancer Splus, lire la fonction avec **source** :

```
> fonc1<-source("fonc1.S")
> fonc1
function(x1, y1)
{
      z1 = x1 * y1
      return(z1)
}
```

Lancer l'exécution avec deux arguments :

```
> fonc1(2,3)
[1] 6
>
```

Le langage S est récursif :

```
> fact<-source("fact.S")
function (x)
{
      if (x > 1) return (x * fact(x - 1))
      else return (x)
}
> fact(8)
[1] 40320
>
```